

## REMARKS

Claims 1-33 are pending in the present application. Replacement Figure 2, denoted by "REPLACEMENT SHEET," is submitted to include the words "PRIOR ART", as suggested by the Examiner. No new matter is added as a result of the replacement. Reconsideration of the claims in view of the following Remarks is respectfully requested.

**I. 35 U.S.C. § 103(a), Alleged Obviousness, Claims 1, 11, 21 and 31-33**

The Office Action rejects claims 1, 11, 21 and 31-33 under 35 U.S.C. § 103(a) as being allegedly anticipated by the Applicants Admitted Prior Art ("Applicant, Specification, pages 1-4) further in view of McGarvey (U.S. Patent No. 5,926,631). This rejection is respectfully traversed.

As to claims 1, 11, 21 and 31-33, the Office Action states:

As per independent claim 1, the Applicant discloses a method in a computer system, said method comprising the steps of:

- a. executing a UNIX-based operating system within said computer system (figure 2, element 202);
- b. executing a Java desktop within said UNIX-based operating system (figure 3, element 304);
- c. executing a window manager proxy within said UNIX-based operating system (figure 2, element 206);

The Applicant does not disclose:

- d. graphically presenting native Java applications within said computer system utilizing a graphical user interface;
- e. and graphically presenting native UNIX applications within said computer system utilizing said graphical user interface, wherein Java applications and UNIX applications are presented by said computer system utilizing the same graphical user interface.

McGarvey teaches to graphically present native Java applications within said computer system utilizing a graphical user interface (column 4, lines 36-38); and to graphically present native UNIX applications within lines 55-60), wherein Java applications and UNIX applications are presented by said computer system utilizing the same graphical user interface (column 10, lines 46-50).

It would have been obvious to one of ordinary skill in the art at the time the invention was made to modify the teachings of the Applicant with the teachings of McGarvey to include a method to graphically present

native Java applications within said computer system utilizing a graphical user interface; and to graphically present native UNIX applications within said computer system utilizing said graphical user interface, wherein Java applications and UNIX applications are presented by said computer system utilizing the same graphical user interface with the motivation to allow the user the flexibility and efficiency of running both java applications and native applications on the same graphical user interface (McGarvey, column 4, lines 55-60).

Claims 11, 21, 31-33 are individually similar in scope to claims 1, and are therefore rejected under similar rationale.

Office Action dated December 23, 2003, pages 2-4.

Independent claim 1, which is representative of claims 11, 21 and 31-33 with regard to similarly recited subject matter, recites:

1. A method in a computer system, said method comprising the steps of:  
executing a UNIX-based operating system within said computer system;  
executing a Java desktop within said UNIX-based operating system;  
executing a window manager proxy within said UNIX-based operating system;  
graphically presenting native Java applications within said computer system utilizing a graphical user interface; and  
graphically presenting native UNIX applications within said computer system utilizing said graphical user interface, wherein Java applications and UNIX applications are presented by said computer system utilizing the same graphical user interface.  
(emphasis added)

The allegedly admitted prior art on pages 1-4 of the present specification discloses a computer system executing a UNIX-based operating system, a desktop implemented in the C language, a window manager and UNIX applications written for the UNIX-based operating system. Pages 1-4 further describe a computer system executing a UNIX-based operating system and having a Java desktop. However, if a window manager is provided in this computer system, the native window manager and the Java desktop together would present conflicting GUI characteristics to the end user. The graphical environment of a native UNIX application includes an application client window rendered by the native application, and a frame window rendered by a window manager. The window manager handles events related to the frame window, while the native application handles events related to the application client window.

McGarvey is directed to a network computer emulator program that runs a Java runtime environment, a Java desktop and Java applications on a personal computer. Data generated by the Java runtime environment, the Java desktop and the Java applications are stored at a server via a network, rather than in a persistent storage of the personal computer. Data for the Java runtime environment, the Java desktop and the Java applications may also be obtained from the server over the network, rather than from the persistent storage. The persistent storage of the personal computer may be used as a Java code cache that stores the Java runtime environment, the Java desktop and the Java applications that are obtained from the server over the network (Abstract).

Neither the alleged admitted prior art nor McGarvey teach or suggest graphically presenting Java applications and native UNIX applications within a computer system utilizing the same graphical user interface. The allegedly admitted prior art, as described on page 2, lines 14-21 of the current specification, describes a computer system that executes a UNIX based operating system, a desktop implemented in C language, a window manager and UNIX applications written for the UNIX based operating system. However, there is no mention of Java applications running on the same computer system in the allegedly admitted prior art.

The Office Action alleges that McGarvey teaches this feature at column 10, lines 46-50, which reads as follows:

Once the Java desktop application is running, the environment presented to the user may be indistinguishable from that of a network computer. However, the user can still access personal computer functions, for example, by running Windows programs concurrently with the emulator.

In the above section, McGarvey teaches that a user may still access personal computer functions by running the functions concurrently with the network computer emulator. McGarvey does not teach or suggest graphically presenting Java applications and native UNIX applications utilizing the same graphical user interface. There is nothing in McGarvey that teaches or suggests presenting Java applications and native UNIX applications utilizing the same graphical user interface.

At column 8, lines 40-50, McGarvey teaches how the network computer emulator accesses applications, which reads as follows:

Preferably, according to the invention, Java applications access only the Java runtime environment classes, the classes of the extended security layer, certain classes of the Java desktop and the classes the application itself loads from the network. However, the applications are prevented from loading Java classes from the personal computer hard disk. Thus, the network computer emulation environment can be made secure, predictable and centrally administrable (emphasis added).

At column 9, line 60 to column 10, line 30, McGarvey teaches the use of Windows programs in the network computer emulator environment, which reads as follows:

Thus, Windows programs can concurrently run with a network computer emulator, using local code and data, without requiring Winframe or other remote application servers. In particular, the user may continue to use existing Windows desktop applications for personal computer intensive functions. However, as more and more functionality is delivered via the network computer emulator environment, the user may have less need to maintain a Windows desktop, and change a Windows desktop and the system administrator may have smaller requirements to maintain applications residing on the user machine. Most, if not all, access to enterprise data can occur through the network computer emulator.

Network computer emulation, according to the present invention, is significantly different from Windows web browser access to Java. In particular, when a user accesses Java applet via a browser, the user installs and maintains the browser and the local Java virtual machine. Many browsers are increasing in functionality and complexity so that maintenance and customization may become more difficult. Moreover, browsers store user preferences and data on local hard disk, so that customization and data generally is local to the machine. Moreover, the user has a responsibility for maintaining data on the local personal computer, for backing it up and for keeping it secure.

The Java support provided by different browsers can vary widely. As a result, many of the Java applets accessible from a network computer may not run under a given browser. Finally, using a Windows browser, the user generally does not go through a standard logon procedure to authenticate the user and to access the network resources. Thus, browser access to Java applets may not provide as much network computer value as the network computer computing. (emphasis added)

In the first section, McGarvey teaches that under the network computer emulator environment, applications are accessed from the network and are prevented from loading from the personal computer's local hard disk. In the second section, McGarvey teaches that the Windows programs may be concurrently run under the network computer emulator environment using local code and data, which are accessed from the personal

computer's local hard disk. McGarvey further teaches that the user may use existing Windows desktop applications to perform intensive functions, but may find less and less need to maintain a Windows desktop as more and more functions are delivered by the network computer emulator. In addition, McGarvey teaches that the network computer emulator is significantly different from a Windows browser access to Java in that the Windows browser requires the user to install and maintain settings on the local hard drive, which makes customization and maintenance more difficult for the user. Furthermore, the Windows browser does not provide the same security features as the network computer emulator.

Based on McGarvey's teachings in the above sections, the network computer emulator supports concurrent execution of Windows programs not by utilizing the same graphical user interface, because 1) the network computer emulator utilizes Java desktop classes that are loaded from the network computer, not from local data and code, which the Windows programs uses; 2) a Windows desktop is required in order to concurrently run Windows programs, as opposed to a Java desktop; 3) the Windows desktop maintains applications on the local hard disk, which is prevented by the network computer emulator; 4) the network computer emulator is significantly different from the Windows web browser based on where user preferences and data are maintained. Therefore, the network computer emulator of McGarvey does not, and would not, present both the Java applications and Windows programs in the same graphical user interface, because of the above substantial differences between the two types of applications. In addition, based on the significant advantages of the network computer emulator that McGarvey teaches over the use of local Windows programs, McGarvey does not and would not teach running both the Windows program and the Java applications utilizing the same graphical user interface. Thus, McGarvey does not teach the features of claim 1.

The Office Action also alleges that it would have been obvious to one of ordinary skill in the art to modify the teachings of the allegedly admitted prior art with the teachings of McGarvey to include a method to present Java applications and native UNIX applications utilizing the same graphical user interface with the motivation to allow the user the flexibility and efficiency of running both Java and native applications on the

same graphical user interface. This motivation is allegedly suggested by McGarvey at column 4, lines 55-60, which reads as follows:

The invention can also preserve some of the advantages of personal computers such as user access to the large base of personal computer native applications, a fast start-up process, reduced loading of the network and access to virtual memory as well as physical memory.

In the above section, McGarvey only suggests a motivation to preserve user access to native applications along with other advantages of the personal computer, such as fast start-up process and reduced loading of network. McGarvey does not suggest any motivation to graphically present Java applications and native UNIX applications utilizing the same graphical user interface. McGarvey is only concerned with using the network computer emulator to run Java applications and Java desktop loaded from a network computer on a personal computer, at the same time allowing the user of the personal computer to access local resources through concurrent execution of Windows programs. As described above, McGarvey teaches using a Windows desktop to accesses applications from the local hard disk of the user, as opposed to a Java desktop that accesses classes from the network computer. In addition, there is no teaching or suggestion in the reference that both the Java desktop and the Windows desktop should be concurrently running utilizing the same graphical user interface, due to the significant differences between the Java applications and the Windows programs, as taught by McGarvey. Therefore, without the Applicants' disclosure, a person of ordinary skill in the art would not be motivated to modify the allegedly admitted prior art or McGarvey to present the native applications and the Java applications utilizing the same graphical user interface.

In view of the above, Applicants respectfully submit that neither the prior art admitted by the Applicants nor McGarvey, either alone or in combination, teach or suggest all of the features of independent claim 1. The other independent claims 11, 21 and 31-33 recite similar features also not taught or suggested by the admitted prior art or McGarvey. Accordingly, Applicants respectfully request the withdrawal of the rejection of claims 1, 11, 21 and 31-33 under 35 U.S.C. § 103(a).

## II. 35 U.S.C. § 103(a), Alleged Obviousness, Claims 2, 6, 16, 22 and 26

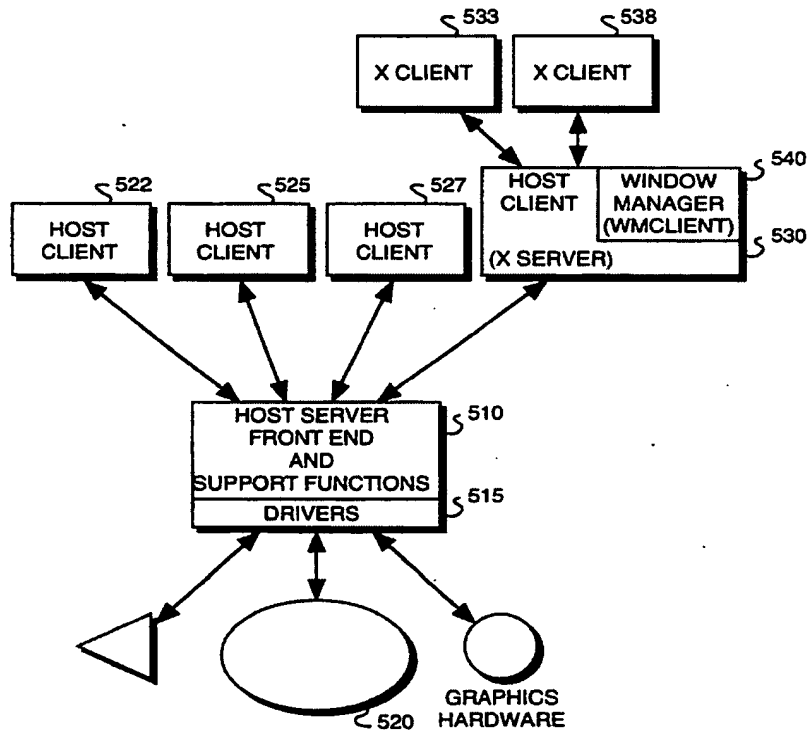
The Office Action rejects claims 2, 6, 16, 22 and 26 under 35 U.S.C. § 103(a) as being allegedly unpatentable over the Applicants Admitted Prior Art ("Applicant, Specification, pages 1-4) and McGarvey (U.S. Patent No. 5,926,631) and further in view of Giokas et al. (U.S. Patent No. 5,408,602). This rejection is respectfully traversed.

As described above, Applicant's allegedly admitted prior art and McGarvey do not teach graphically presenting both the Java applications and the native UNIX applications utilizing the same graphical user interface, as recited in claim 1. Giokas also does not teach these features. Giokas teaches an X window display server that provides a virtual window manager client that, from a view point of client programs connected to the server, is indistinguishable from a real window manager client. The emulated window manager is implemented as an internal server client (Abstract). Giokas teaches emulating the presence and functionality of a X window manager client within the X display server. Giokas does not mention anything about running both Java applications and native UNIX applications utilizing the same graphical user interface. Giokas only teaches how a window manager client may be implemented in a server computer, Giokas does not teach presenting both the Java applications and the native UNIX Applications utilizing the same graphical user interface.

As to dependent claim 2, which is representative of claims 12 and 22 with regard to similarly recited subject matter, the Office Action alleges that Giokas teaches distributing window manager functions between a Java desktop and a window manager proxy, at column 6, lines 46-51, which reads as follows:

In the preferred embodiment of the invention shown in **FIG. 5**, the host server **510**, has internal window management functions. WMCLIENT **540** takes advantage of this implementation of window management functions inside of the host server **510** by delegating window management functions to the host server **510**.

In the above section, Giokas teaches that WMCLIENT **540** on X display server **530** delegates window management functions to host server **510**, so that the WMCLIENT may still provide window management functions to the X clients without implementing the functions within it. **Figure 5** of Giokas is shown below:



As shown in **Figure 5**, Giokas teaches WMCLIENT 540, which is an emulation of the window manager client, is implemented within X display server 530. The emulation is invisible to X clients 533 and 538. Generally, the X window clients send requests to the X Window server to perform graphic functions. The X Window server in turn generates events to inform clients of the status of a window. Giokas teaches that instead of having WMCLIENT 540 on X display server 530 implementing window management functions, WMCLIENT 540 may delegate the functions to host server 510, so that the request may be deferred to host server 510. However, Giokas does not teach distributing window manager functions between a Java desktop and a window manager proxy, as recited in claim 2.

When read in combination with claim 1, both the Java desktop and the window manager proxy recited in claim 2 are executed within the same UNIX-based operating system of the same computer system. Thus, the window management functions are distributed among the Java desktop and the window manager proxy of the same computer



system. To the contrary, in the section above, Giokas teaches sending graphic function requests from X clients to an X display server, which in turn delegates window management functions to a host server. The host server then provides a common window interface that is the same window interface as other host clients to the X clients. Thus, Giokas teaches a different way of supplying window management functions than the presently claimed invention in that Giokas teaches delegating window management functions to a server system separate from a computer system. This is contrary to the presently claimed invention, which distributes window management functions between the Java desktop and the window manager proxy. Both of which are executed within the same computer system. Therefore, Giokas does not teach the features of claim 2.

The Office Action further alleges that it would have been obvious to one of ordinary skill in the art to modify the teachings of the allegedly admitted prior art and McGarvey to distribute window management functions between more than one window management tool, with the motivation to distribute the processing of windowing tasks and to maintain a common look and feel to the interface. The motivation is allegedly suggested by Giokas at column 6, lines 51-58, which reads as follows:

This delegation has two principal advantages. First, WMCLIENT can supply window management functions to X clients without implementing those functions within WMCLIENT itself. Second, X client managed windows inherit windowing interfaces of the PM system so that X clients will have a common window interface with host clients ("PM clients").

Applicants respectfully disagree with the Examiner. As described above, the host server and WMCLIENT of Giokas are executed on an X display server, which is a separate system from the computer system. In addition, in the above section, Giokas only teaches supplying window management functions in a host server instead of the X display server, so that the X client may inherit a common window interface that is the same window interface as other host clients. Thus, Giokas only teaches a single host server that provides window management functions. Giokas does not give any incentive to distribute the window management functions between more than one window management tool, meaning a Java desktop and a window manager proxy.

In addition, neither the allegedly admitted prior art, McGarvey nor Giokas, either alone or in combination, teach or suggest a window manager proxy, which implements a

subset of functions of a full-featured window manager. The admitted prior art only teaches a window manager, not a window manager proxy. McGarvey only teaches a network computer emulator that runs Java desktop, Java runtime environment and Java applications on a personal computer, while allowing user to access native applications within the personal computer. Giokas only teaches emulating a window manager within an X display server. There is no mention of a window manager proxy, which includes a subset of services of the full-featured window manager.

Therefore, it would not be obvious for a person of ordinary skill in the art to modify the admitted prior art and McGarvey to distribute window management functions between more than one window management tool in order to distribute processing of windowing tasks and to maintain a common look and feel to the interface, because none of the references teach or suggest a window manager proxy or distributing window management functions between a Java desktop and a window manager proxy.

As to dependent claim 6, which is representative of claims 16 and 26 with regard to similarly recited subject matter, Giokas does not teach establishing a communication interface support within a window manager proxy for permitting applications to connect to and interact with the window manager proxy, routing a first plurality of events utilizing the window manager proxy to the Java desktop for processing, or processing a second plurality of events by the window manager proxy. The Office Action alleges that Giokas teaches these features at column 7, lines 1-5, which reads as follows:

WMCLIENT 540 learns of the results of a user's manipulation of these objects by receiving PM events. For example, a user may select a button to move a window, resulting in WMCLIENT 540 receiving a PM move window event, WM\_MOVE.

In the above section, Giokas teaches that a WMCLIENT, which resides on a X display server, receives a presentation manager (PM) move window event from a client when a user selects a button to move a window. Giokas only teaches receiving PM events from a client as the client manipulates the window interface. However, nowhere in the above section, or any other section, of the reference does Giokas teach or suggest a communication interface support within a window manager proxy for permitting

applications to connect to and interact with the window manager proxy, as recited in claim 6.

As described above, the window manager proxy of the present invention is different from a window manager in that the window manager proxy only includes a subset of services, that is, the message passing functions between applications and Java desktop, of a full-featured window manager. As described on page 5 of the current specification, the window manager proxy forwards frame window activity related to the desktop to the Java desktop and is the native interface from the Java desktop to the native platform for frame window activities. Giokas does not teach such window manager proxy. Giokas only teaches a host server that provides window management functions when delegated by the WMCLIENT of the display server. Neither the host server nor the WMCLIENT of the X display server is the same as a window manager proxy.

In addition, Giokas does not teach a communication support interface that permits applications to connect and interact with the window manager proxy. In the above section, Giokas only teaches permitting the WMCLIENT of the X display server to receive events from a client. Giokas does not teach permitting client applications to connect and interact with the window manager proxy. The main difference between a window manager proxy and the WMCLIENT of the X display server is that the window manager proxy and the applications are both executed on the same UNIX-based operating system of the same computer system. The WMCLIENT of the X display server is a separate system from the computer system. Thus, Giokas teaches a communication support interface that permits a display server to interact with the client. Giokas does not teach a communication interface that permits applications and a window manager proxy, both residing on the same computer system, to connect and interact with each other.

Furthermore, Giokas does not teach routing a first plurality of events utilizing the window manager proxy to the Java desktop for processing, or processing a second plurality of events by the window manager proxy. As described above, Giokas does not teach a window manager proxy running on a client computer system, or a Java desktop running on the same client computer system. Giokas only teaches routing a first plurality of events generated by a client to WMCLIENT of the X display server, which is a separate computer system from the client, for processing. There is no utilization of any

window manager proxy, since Giokas fails to teach a window manager proxy. Therefore, Giokas does not teach routing a first plurality of events utilizing a window manager proxy to the Java desktop for processing.

Moreover, Giokas does not teach processing a second plurality of events by the window manager proxy. Since Giokas does not teach a window manager proxy, Giokas would not teach processing a second plurality of events by the window manager proxy. Thus, Giokas does not teach the features of claim 6, as alleged in the Office Action.

In view of the above, Applicants respectfully submit that neither the prior art admitted by the Applicants, McGarvey nor Giokas, either alone or in combination, teach or suggest the specific features of dependent claims 2 and 6. The other dependent claims 12, 16, 22 and 26 recite similar features also not taught or suggested by the admitted prior art, McGarvey or Giokas. Accordingly, Applicants respectfully request the withdrawal of the rejection of claims 2, 6, 12, 16, 22 and 26 under 35 U.S.C. § 103(a).

### **III. 35 U.S.C. § 103, Alleged Obviousness, Claims 10, 20 and 30**

The Office Action rejects claims 10, 20 and 30 under 35 U.S.C. § 103 as being allegedly unpatentable over the Applicants Admitted Prior Art (“Applicant, Specification, pages 1-4) and McGarvey (U.S. Patent No. 5,926,631) and further in view of Matthews et al. (U.S. Patent No. 5,974,256). This rejection is respectfully traversed.

As described above, neither the allegedly admitted prior art nor McGarvey, either alone or in combination, teach or suggest presenting both the Java applications and native UNIX applications utilizing the same user interface, Matthews also does not teach these features. Matthews is directed to a method of translating layout defined by a resource definition file having a set of resources directive associated to native Java source code. The method includes identifying each resource file directive associated with the window layout. For each directive or subdirective, the method generates a stub or “snippet” of Java code. The Java code snippet may be generated by direct code or using a text-editable file interface, and the snippets are then combined to create a Java source file (Abstract). Thus, Matthews only teaches how source code may be translated from a set

of resource directives to a Java source file. Matthews does not teach presenting either or both of the source files utilizing the same graphical user interface.

With regard to claims 10, 20 and 30, Applicants agree with the Examiner that the allegedly admitted prior art and McGarvey does not teach or suggest intercepting from one of the native UNIX applications, utilizing a window manager proxy, a frame window event to a Java Native Interface, transmitting the translated frame window event to a Java desktop, or executing the translated frame window event utilizing the Java desktop to render a new window. However, Applicants respectfully disagree with the Examiner that Matthews teaches these features at column 7, lines 60-65, column 8, lines 57-60, and column 9, lines 57-65, which reads as follows:

Once the resource data is transferred into Java native code that makes up the windowing framework, the event management and processing code must be added or merged into the Java framework to produce a working application. The event management code listens for an action by the user, e.g., a button push, menu item, list box, and responds with the appropriate processing code. The processing code is where the work is done (worker code) and is generally unique for each event.

(Column 7, lines 60-65, Matthews)

Below is an example of a menu event and processing code for Java 1.02; those skilled in the art would understand that Java 1.1 and higher event delegation modules could be used. Further, native event management and processing code could be integrated into the Java framework, e.g., C or C++, if the Java to native interface was utilized.

(Column 8, lines 57-60, Matthews)

This code may then be used to port the native-based windowing layout to Java. In particular, appropriate event management code may be incorporated into the Java code to provide a Java-based version of the interface. With this backbone, an existing application may then be readily ported to Java code in a prompt and efficient manner.

Thus, after the skeletal structure of the windowing layout has been created according to the present invention, the basic “look and feel” of the GUI exists in Java. A set of “work” code must still be incorporated in order to provide the working interface.

(Column 9, lines 57-65, Matthews)

In the above sections, Matthews teaches a translation tool that translates a window layout from a native directive resource file to Java native code, in order to facilitate migration of existing native applications to Java. Matthews further teaches that, with a

skeletal structure of the windowing layout provided by the translation tool, event management code may be incorporated into the Java native code to provide a complete working application. However, Matthews does not teach intercepting one of the native UNIX applications, utilizing the window manager proxy, a frame window event to render a new window. Matthews does not teach anything about a window manager proxy, which includes a subset of functions of a full-featured window manager. Matthews only teaches a translation tool for translating a window layout defined using resource directives of a resource definition file to Java native code. Since Matthews does not teach a window manager proxy, Matthews would not teach utilizing the window manager proxy to intercept a frame window event to render a new window.

In addition, Matthews does not teach forwarding the frame window event to a Java native interface utilizing the window manager proxy. As described above, no window manager proxy is taught or suggested by Matthews. In the above sections, Matthews merely teaches integrating C or C++ native event processing code into the Java framework if a Java to native interface is utilized. Thus, Matthews teaches integrating native event processing code, in C or C++, with the Java native interface to generate Java native code. Matthew does not mention anything about forwarding a frame window event to render a new window to a Java native interface utilizing a window manager proxy.

The Office Action alleges that it would have been obvious for a person of ordinary skill in the art to modify the teachings of the combination of the allegedly admitted prior art and McGarvey with a means to intercept window event, translate these events using the Java Native Interface, and render windows on the Java Desktop, as taught by Matthews, with the motivation to allow platform independent user interaction with frame windows. This motivation is allegedly suggested by Matthews at column 1, lines 61-63, which reads as follows:

It is a by-product or advantage of this invention to reduce the time and effort necessary to port a platform dependent software application into a platform-independent programming environment such as Java.

Matthews teaches a motivation to translate platform dependent applications from an event driven windowing environment, which uses resource definition files, to a

platform independent Java native code for the purpose of reducing time and effort required for a migration. Matthews is not concerned with using a window manager proxy to intercept frame window event initiated from a native UNIX application to a Java desktop, all of which are running on the same computer system, in order to translate a new window event from a C language to a Java language understandable by the Java desktop for the purpose of rendering a new window. Matthews is only concerned with translating windowing layout of an event driven based application to Java native code, not frame window event that handles actions initiated to the windowing layout.

Matthews actually requires a user to add or merge event-processing code to the Java framework in order to produce a working application that handles the events to the windowing layout. Therefore, Matthews does not teach intercepting or translating a frame window event in order to render a new window.

Thus, a person of ordinary skill in the art would not be motivated to modify the teachings of the allegedly admitted prior art and McGarvey to include Matthews's teachings to intercept and translate frame window events using a window manager proxy in order to render a new window, because Matthews does not teach anything about intercepting the frame window event, nor does Matthews teach or suggest anything about translating the frame window event for the purpose of rendering a new window.

In view of the above, Applicants respectfully submit that neither the allegedly admitted prior art, McGarvey nor Giokas, either alone or in combination, teach or suggest the specific features of dependent claim 10. The other dependent claims 20 and 30 recite similar features also not taught or suggested by the admitted prior art, McGarvey or Giokas. Accordingly, Applicants respectfully request the withdrawal of the rejection of claims 10, 20 and 30 under 35 U.S.C. § 103(a).

**IV. 35 U.S.C. § 103(a), Alleged Obviousness, Claims 3-5, 7-9, 13-15, 17-19, 23-25 and 27-29**

The Office Action rejects claims 3-5, 7-9, 13-15, 17-19, 23-25 and 27-29 under 35 U.S.C. § 103(a) as being allegedly unpatentable over the Applicants Admitted Prior Art (“Applicant, Specification, pages 1-4), McGarvey (U.S. Patent No. 5,926,631), and Giokas et al. (U.S. Patent No. 5,408,602), and further in view of Matthews et al. (U.S. Patent No. 5,974,256). This rejection is respectfully traversed.

As described above, neither the allegedly admitted prior art, McGarvey, Giokas, or Matthews, either alone or in combination, teach or suggest presenting both the Java applications and the native UNIX applications utilizing the same user interface. In addition, with regard to claims 3, 13 and 23, neither the admitted prior art, McGarvey, Giokas nor Matthews teach or suggest utilizing, by the Java desktop, the window manager proxy to communicate with the native UNIX applications. The Office Action alleges that Matthews teaches these features at column 8, lines 56-60, which is reproduced above, column 2, lines 13-15 and 14-18, which reads as follows:

After the directives are processed, the basic “skeletal” structure of the windowing layout is generated in native Java source code. Appropriate event management code may then be incorporated into the Java code to provide a Java-based version of the interface. With this backbone, the enterprise may port its existing application to Java in a prompt and efficient manner.

In the above section, Matthews only teaches generating windowing layout in native Java source code by processing resource directives in a resource definition file. Event management code may be incorporated into the Java native code to provide a Java based interface with the same windowing layout. However, Matthews does not teach utilizing the window manager proxy to communicate with the native UNIX applications by the Java desktop. As described above, nowhere in the reference does Matthews teach or suggest a window manager proxy. In addition, Matthews only teaches translating windowing layout from resource directives to Java native code, Matthews does not teach using the translation tool to facilitate communications between the Java desktop and a native UNIX application. Actually, with the teaching of Matthews, there will not be a need to facilitate communications between the Java desktop and the native UNIX application, because the windowing layout has been translated to Java native code from a



native format for migration. Therefore, Matthews does not and would not teach the features of claims 3, 13 and 23.

The Office Action further alleges that it would have been obvious to one of ordinary skill in the art to modify the teachings of the combination of the allegedly admitted prior art, McGarvey and Giokas with a means to manage user interactions with frame windows using the Java desktop and a window manager proxy, as taught by Matthews, with the motivation to allow platform dependent user interaction with frame window. This motivation is allegedly suggested by Matthews at column 1, lines 61-63, which is reproduced above. However, as stated in the above section, the motivation of Matthews is to port existing applications to Java in a prompt and efficient manner. The motivation of Matthews is not to allow platform dependent user to interact with frame window. Matthews is not concerned with using a window manager proxy to facilitate communications between the Java desktop and the native UNIX application for the purpose of managing frame windows. Therefore, a person of ordinary skill in the art would not be motivated to modify the teachings of the combination of the Applicant, McGarvey and Giokas to allow platform dependent user to interact with frame windows, as alleged in the Office Action.

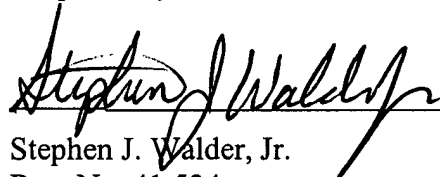
In view of the above, Applicants respectfully submit that neither the allegedly admitted prior art, McGarvey, Giokas nor Matthews, either alone or in combination, teach or suggest the specific features of dependent claims 3-5, 7-9, 13-15, 17-19, 23-25 and 27-29 in addition to their dependency on claims 1, 11 and 21, respectively. Accordingly, Applicants respectfully request the withdrawal of the rejection of claims 3-5, 7-9, 13-15, 17-19, 23-25 and 27-29 under 35 U.S.C. § 103(a).

V. Conclusion

It is respectfully urged that the subject application is patentable over the admitted prior art of the Applicants, McGarvey, Giokas, and Matthews and is now in condition for allowance. The Examiner is invited to call the undersigned at the below-listed telephone number if in the opinion of the Examiner such a telephone conference would expedite or aid the prosecution and examination of this application.

DATE: March 23, 2004

Respectfully submitted,



Stephen J. Walder, Jr.  
Reg. No. 41,534  
Carstens, Yee & Cahoon, LLP  
P.O. Box 802334  
Dallas, TX 75380  
(972) 367-2001  
Attorney for Applicants

SJW/im